AP-
IPW

20IN THE UNITED STATES PATENT AND TRADEMARK OFFICE BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

Application No.: 09/886,455
Filed: June 20, 2001
Inventor(s):
Jeffrey D. Washington, Ram Kudukoli, Robert E. Dye, and Paul F. Austin

Title: System and Method for Programmatically Generating a Graphical Program in Response to User Input

§ § § § § § § § § § § § § §

Examiner: Vu, Kieu D.
Group/Art Unit: 2173
Atty. Dkt. No: 5150-48300

I hereby certify that this correspondence is being deposited with the United States Postal Service with sufficient postage as first class mail in an envelope addressed to Commissioner for Patents, Alexandria, VA 22313-1450, on the date indicated below.

Jeffrey C. Hood

[illegible] 11/15/2004
Signature Date

APPEAL BRIEF

Box: Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir/Madam:

Further to the Notice of Appeal filed October 12, 2005, Appellant presents this Appeal Brief. Appellant respectfully requests that this appeal be considered by the Board of Patent Appeals and Interferences.

I. REAL PARTY IN INTEREST

The subject application is owned by National Instruments Corporation, a corporation organized and existing under and by virtue of the laws of the State of Delaware, and having its principal place of business at 11500 N. MoPac Expressway, Bldg. B, Austin, Texas 78759-3504.

II. RELATED APPEALS AND INTERFERENCES

No related appeals or interferences are known which would directly affect or be directly affected by or have a bearing on the Board's decision in this appeal.

III. STATUS OF CLAIMS

Claims 1 – 51 were originally filed in the application. In an amendment filed July 23, 2004, claims 1, 16, 17, 18, 21, 23, 26, and 34 were amended, claims 9, 24, 25, 27, 28, 42, 47, and 48 were cancelled, and new claims 52 and 53 were added. In an amendment filed April 7, 2005, claims 1, 6, 7, 10-12, 14-18, 21-23, 26, 29-34, 40, 43, 44, 46, and 49-51 were amended, and claims 8, 39, and 41 were cancelled. Thus, claims 1-7, 10-23, 26, 29-38, 40, 43-46, and 49-53 are pending in the case.

Claims 1-7, 10-23, 26, 29-38, 40, 43-46, and 49-53 stand rejected under 35 U.S.C. § 103(a) and are the subject of this appeal. A copy of claims 1-53 incorporating entered amendments, including claims 1-7, 10-23, 26, 29-38, 40, 43-46, and 49-53 as on appeal, is included in the Claims Appendix hereto.

IV. STATUS OF AMENDMENTS

No amendments to the claims have been filed subsequent to the rejection in the Office Action of July 14, 2005. The Claims Appendix hereto reflects the current state of the claims.

V. SUMMARY OF THE CLAIMED SUBJECT MATTER

The present application relates generally to the field of graphical programming, and more particularly to a system and method for programmatically or automatically generating a graphical program in response to user input specifying functionality of the graphical program.

Independent claim 1 is directed to a computer-implemented method for programmatically generating a graphical program. A graphical user interface (GUI) is displayed on a display, e.g., a computer display screen or monitor. For example, the GUI may include one or more input panels that include various GUI input controls such as text boxes, check boxes, list controls, numeric controls, etc., that the user may configure to indicate the desired program functionality. *See, e.g., Specification, p. 34:4-14, p. 16:9-12, p. 35:10-19; Figure 6:300.* User input is received to the GUI specifying desired functionality of the graphical program. *See, e.g., Specification, p. 34:15-25, p. 35:10-19; Figure 6:302.* A graphical program implementing the specified functionality is automatically generated in response to the user input specifying the functionality of the graphical program. *See, e.g., Specification, p. 34:26 – p.35:9; Figure 6:304.* The graphical program includes a block diagram portion comprising a plurality of interconnected nodes, and a graphical user interface portion, where the plurality of interconnected nodes visually indicate functionality of the graphical program. *See, e.g., Specification, p. 7:13-20, p. 24:9-20; Figures 9-12.* Automatically generating the graphical program includes generating the block diagram portion without direct user input specifying the plurality of nodes or connections between the nodes. In other words, user input selecting, arranging, and interconnecting the nodes in the new graphical program is not required. *See, e.g., Specification, p. 22:11-16.*

Independent claim 16 is directed to a computer-implemented method for programmatically generating a graphical program similar to that of claim 1, but where the GUI is specified as including a plurality of GUI input panels, where the plurality of GUI input panels are displayed on a display, and include information useable in guiding a user in creation of a program. User input specifying desired functionality of the graphical program is received to the plurality of GUI input panels. *See, e.g., Specification, p. 35:10-19.* The new graphical program is then generated automatically, as described above with reference to independent claim 1.

Independent claim 17 is directed to a computer-implemented method for automatically generating a graphical program similar to that of independent claim 1, but where user input specifying desired functionality of the graphical program is provided to a graphical program generation (GPG) program, which then automatically generates the graphical program, as described above with reference to independent claim 1. *See, e.g., Specification, p. 22:17 – p. 33:5; Figure 5.*

Independent claim 18 is directed to a computer-implemented method for automatically generating a graphical program similar to that of independent claim 1, but where user input indicating desired program operation is provided to a graphical program generation (GPG) program, which then automatically generates the graphical program, as described above with reference to independent claim 1. *See, e.g., Specification, p. 22:17 – p. 33:5; Figure 5.*

Independent claim 21 is directed to a computer-implemented method for automatically generating a graphical program. One or more input panels are displayed on a display, e.g., on a computer display, and user input received to the one or more input panels. *See, e.g., Specification, p. 34:4-25, p. 16:9-12, p. 35:10-19; Figure 6:300 and 302.* Graphical source code for the graphical program is automatically generated based on the received user input. *See, e.g., Specification, p. 24:1-4, p. 27:7-11, p. 34:26 – p.35:9; Figure 6:304.* The graphical program includes a block diagram portion comprising a plurality of interconnected nodes, and a graphical user interface portion, wherein the plurality of interconnected nodes visually indicate functionality of the graphical program. *See, e.g., Specification, p. 7:13-20, p. 24:9-20; Figures 9-12.* Automatically generating graphical source code for the graphical program includes generating graphical source code for the block diagram portion without direct user input specifying the plurality of nodes or connections between the nodes. *See, e.g., Specification, p. 22:11-16.*

Independent claim 23 is directed to a computer-implemented method for automatically generating a graphical program. A node is displayed in the graphical program in response to user input. *See, e.g., Specification, p. 34:11-12, p. p. 35:27 – p. 36:5; Figure 7:310.* A graphical user interface (GUI) for configuring functionality for the node is displayed in response to user input, e.g., in response to user input requesting to specify desired functionality or configuration information for the node. For example, the user may double click on the node, execute a menu option for configuring the node, or perform this request in any of various other ways. *See, e.g., Specification, p. 36:6-12; Figure 7:312-314.* User input is be received via the GUI indicating desired functionality for the node. *See, e.g., Specification, p. 36:12-21; Figure 7:316.* Graphical source code associated with the node in the graphical program is then automatically included in the graphical program, where the automatically included graphical source code implements the desired functionality, and where the graphical source code includes a plurality of interconnected nodes that visually represent the desired functionality. *See, e.g., Specification, p. 34:26 – 35:3, p. 36:22 – p. 37:2, p. 24:9-20; Figure 7:318.* Automatically including graphical source code associated with the node in the graphical program includes automatically including the graphical source code as a sub-program of the graphical program without direct user input specifying the plurality of nodes or connections between the nodes, where the node represents the sub-program. *See, e.g., Specification, p. 36:22 – p. 37:15, p. 24:9-20; Figure 7:318. See also, e.g., Specification, p. 39:12 – p. 40:20; Figures 8-14.*

Independent claim 26 is directed to a method for configuring a node in a graphical program similar to that of claim 23, but where the GUI display is particularly associated with the node. *See, e.g., Specification, p. 36:10-12, and the citations indicated above re claim 23.*

Independent claims 34 is a memory medium analogue claim to claim 1.

Independent claims 46 is a memory medium analogue claim to claim 26.

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

Claims 1-7, 10-23, 26, 29-38, 40, 43-46, and 49-53 were rejected under 35 U.S.C. 10 3(a) as being unpatentable over U.S. Patent No. 6,366,300 to Ohara et al ("Ohara") and U.S. Patent No. 5,946,485 to Weeren et al. ("Weeren").

VII. ARGUMENT

Section 103(a) Rejection

Claims 1-7, 10-23, 26, 29-38, 40, 43-46, and 49-53 were rejected under 35 U.S.C. 10 3(a) as being unpatentable over U.S. Patent No. 6,366,300 to Ohara et al ("Ohara") and U.S. Patent No. 5,946,485 to Weeren et al. ("Weeren"). Appellant respectfully traverses this rejection for the following reasons. Different groups of claims are addressed under their respective subheadings.

Claims 1, 2, 3, 4, 16, 21, 22, 34, 35, 36, 37

Claim 1 is separately patentable because the cited reference does not teach or suggest the limitations recited in this claim.

The Examiner admits that Ohara fails to teach that the graphical program comprises a user interface portion, as well as automatically generating the block diagram portion without direct user input specifying the plurality of nodes or connections between the nodes, but asserts that Weeren remedies this admitted deficiency. Appellant respectfully disagrees.

For example, the Examiner asserts that Weeren teaches that the (automatically generated) graphical program comprises a graphical user interface portion, citing "VerifyAcct" 402 of Figure 4. However, the cited element (VerifyAcct 402) is not a "graphical user interface portion" at all, but rather is a program icon with multiple returns representing a user-defined subroutine, as clearly described in col. 5:47 – col. 6:4. In fact, the only mention Weeren makes of a graphical user interface (GUI) at all is regarding a GUI for the operating system (col. 4:47-49). Thus, Weeren (and Ohara) fails to disclose this feature of claim 1. The Examiner also asserts that Weeren discloses automatically generating the block diagram portion without direct user input specifying the plurality of nodes or connections between the nodes, citing Figure 6 and col.7:20-30. Appellant respectfully submits that Weeren discloses automatically drawing "arrows" (i.e., connections) associated with new program icons added to the program flow by the user, but does not teach or suggest automatically generating the block diagram portion *without direct user input specifying the plurality of nodes.* For example, the cited text reads:

> Flowchart 600 in FIG. 6 illustrates the steps performed in an exemplary embodiment of the present invention. In step 601, a program flow is graphically represented as a plurality of icons connected by at least one arrow. Additional icons are added to the program flow in step 602. The added icons represent a subroutine having overwriteable components that have a functionality which is represented by second arrows that are added in step 603. Other icons can be added to the second arrows in step 604. In step 605, the graphical development program indicates whether the added subroutine has been overwritten.

Note that no mention is made of automatically selecting or specifying icons to add to the flow diagram. In fact, as described in col.2:47-59:

> ***A developer adds functionality to the program flow by adding icons on existing arrows.*** The graphical development environment automatically adjusts the program flow to include the new icon and its associated arrows.
>
> For example, if ***the developer drops an icon representing a loop somewhere between the start and end of the program flow***, the environment automatically inserts the icon into the flow at that point and draws an arrow representing the loop. Subsequently, ***the developer can add other language components within the loop merely by dropping those icons on the arrow representing the loop.*** (*emphasis added*)

Thus, in Weeren's system, icons added to the program flow diagram are specifically selected by the user and added/dropped by the user onto arrows in the diagram. In other words, according to Weeren, the user specifically provides direct user input specifying the program icons to be added to the program flow diagram. Thus, Weeren (and Ohara, as admitted by the Examiner) fails to teach or suggest these features and limitations of claim 1.

Furthermore, Appellant notes that Ohara explicitly excludes automatic generation of a program from the description of Ohara's invention:

> "In the following description, only matters related to the user interface characterizing the visual programming method are explained, excluding automatic generation of a program ***because the automatic generation of a program is the same as the conventional system***."(col. 16, lines 26-28)

In other words, Ohara does not consider the automatic generation of the program in Ohara's system and method to be novel, and does not disclose how it is performed, other than to say it is performed according to prior art approaches. Moreover, the generated program of Ohara is never described as a graphical program comprising a plurality of interconnected nodes that visually represent the desired functionality of the program. In fact, Appellant notes that Ohara never even mentions a graphical program at all, but rather discloses graphical *programming*, meaning creating a program via graphical means, i.e., via a graphical user interface without hand-coding the program, which, in light of Ohara's silence regarding graphical programs, as well as Ohara's statement that the actual automatic generation of the program is the same as the conventional system, indicates that Ohara's generated program (for deployment to the PLC) is a text-based program.

Thus, Appellant further submits that Ohara fails to properly disclose automatic generation of a graphical program, and thus does not provide proper support for the 103 rejection. Similarly, as argued above, Weeren also fails to disclose the automatic generation of a graphical program as claimed, and so also does not provide proper support for the 103 rejection.

Moreover, Appellant submits that neither Ohara or Weeren provides a proper motivation to combine. Appellant notes that the motivation to combine Ohara and Weeren suggested by the Examiner, "so that a program can be automatically generated without user's manual connections between nodes" (citing col. 2:17-21) is simply a statement of perceived benefit of the alleged combination. For example, the cited text reads:

> Accordingly, there is a need in the art for a graphical development environment that automatically connects icons in accordance with the call flow functionality of the underlying language component.

Nowhere does Weeren suggest the desirability of automatically generating the nodes (icons) in the graphical program, i.e., without direct user input specifying the plurality of nodes or connections between the nodes. Nor does Ohara suggest the desirability of automatically generating a user interface portion of a graphical program, nor of automatically generating the block diagram portion without direct user input specifying the plurality of nodes or connections between the nodes. Thus, Appellant submits that the Examiner's attempt to combine Ohara and Weeren to make a section 103 rejection is improper.

Additionally, Appellant submits that even were Ohara and Weeren properly combinable, which Appellant argues they are not, the resulting combination would still not produce Appellant's invention as represented in claim 1.

Thus, for at least these reasons, Appellant submits that Ohara and Weeren, taken singly or in combination, fail to teach or suggest all the features and limitations of claim 1. Appellant notes that the above arguments are also applicable to the other independent claims grouped therewith.

Claims 5, 38

In addition to the distinctions noted above with regard to claim 1, the cited art fails to teach "wherein the second GUI input panel is one of a plurality of possible second GUI input panels, wherein the second GUI input panel is displayed based on the first user input".

In asserting that Ohara teaches these features, the Examiner cites the "next operation" button in Figure 4, which activation results in the display of Figure 6. However, Appellant respectfully submits that the display of Figure 6 is not presented as one of a plurality of possible panels, but rather, is particularly invoked by the user's activation of the button. In other words, in Ohara's system, there is a single display window (shown in Figure 6) that is invoked by the next operation button of Figure 4, Additionally, the user selecting the "next operation" button in no way comprises receiving user input to the GUI ***specifying desired functionality of the graphical program***., but rather, is simply a navigation tool for invoking the next display window. In other words, simply pressing a "next operation" button does not in itself specify desired functionality of the graphical program.

Said another way, Ohara fails to disclose plural options as to panel displays where the particular panel displayed is based on user input specifying desired functionality of the graphical program.

Thus, for at least these reasons, Appellant submits that Ohara and Weeren, taken singly or in combination, fail to teach or suggest all the features and limitations of claim 5.

Claims 6

In addition to the distinctions noted above with regard to claim 1, the cited art fails to teach "wherein said automatically generating the graphical program comprises automatically generating a portion of a graphical program".

In asserting that Ohara teaches these features, the Examiner cites Figure 17. However, Appellant notes that Figure 17 is described thusly:

> FIG. 17 is an explanatory diagram showing a portion of the component model window 160. On the component model window 160 shown in FIG. 17, typically, combinations of behavior relevant signals and behavioral characteristics displayed on the behavior verification window 1300 shown in FIG. 14 are expressed globally on a macro basis. Since the purpose of the behavior verification window 1300 is to locally verify operations carried out by a program generated for an output signal, a logic equation and behavioral characteristics are displayed in detail. On the other hand, the component model window 160 is used for verification of setting of behavioral characteristics for all output signals globally on a macro basis. Pieces of information set for behavioral characteristics of all output signals are thus displayed on the component model window 160 globally on a macro basis in order to allow all the pieces of information to be displayed at the same time. It should be noted that, in the example shown in FIG. 17, information on only one output signal is displayed though. (col. 38:65 – col. 39:16)

Appellant respectfully submits that Ohara nowhere describes Figure 17 as a graphical program, nor as a portion of a graphical program. Rather, as described in the cited text, "the component model window 160 is used for verification of setting of behavioral characteristics for all output signals globally on a macro basis", where "Pieces of information set for behavioral characteristics of all output signals are thus displayed on the component model window 160 globally on a macro basis in order to allow all the pieces of information to be displayed at the same time." Appellant thus respectfully submits that the Examiner's characterization of Figure 17 as a portion of a graphical program is incorrect, and is not supported by the cited text. Moreover, as noted above, Ohara never mentions graphical programs at all. Additionally, as noted above, Ohara's generated program includes no user interface portion, which is a necessary feature of Appellant's invention as claimed. Nor does Weeren teach or suggest these features of claim 6.

Thus, for at least these reasons, Appellant submits that Ohara and Weeren, taken singly or in combination, fail to teach or suggest all the features and limitations of claim 6.

Claims 7, 40

In addition to the distinctions noted above with regard to claim 1, the cited art fails to teach "wherein said automatically generating the graphical program creates the graphical program without any user input specifying the new graphical program during said creating".

In asserting that Ohara teaches these features, the Examiner only cites a "default setting". Appellant notes that Ohara references to default settings relate to default values of specific parameters for various operations, e.g., behavioral characteristics and behavioral conditions, and not to the general specification of the program itself. In other words, Ohara's cited default settings only potentially obviate a subset of the user input specifying the program to be generated. Appellant respectfully submits that Ohara nowhere describes automatically creating a graphical program without *any* direct user input specifying the new graphical program during said creating. Nor does Weeren teach or suggest these features of claim 7.

Thus, for at least these reasons, Appellant submits that Ohara and Weeren, taken singly or in combination, fail to teach or suggest all the features and limitations of claim7.

Claims 10, 43

In addition to the distinctions noted above with regard to claim 1, the cited art fails to teach "automatically creating a plurality of graphical program objects in the graphical program; and automatically interconnecting the plurality of graphical program objects in the graphical program; wherein the interconnected plurality of graphical program objects comprise at least a portion of the graphical program".

In asserting that Ohara teaches these features, the Examiner cites col. 23:1-10, and Figure 17. However, the cited text actually reads:

> The relation is verified starting with a visual object (or an icon) displayed in the component area 163 at a location closest to the input signal area 161. An icon 164 indicates that the behavioral states of the output signal Y000 are on and off states. The icon 164 also represents a logic portion in which the 3 input signals are wired in accordance with a logic equation generated by a logic equation generating means 13. An icon 165 is an icon for expressing a delay behavioral characteristic and an icon 166 expresses a flickering behavioral characteristic.

The cited text relates to verification, not program generation. Nowhere does the cited text (or any other portion of Ohara) disclose automatically creating a plurality of graphical program objects in the graphical program; and automatically interconnecting the plurality of graphical program objects in the graphical program, in the manner claimed, i.e., without direct user input specifying the plurality of nodes or connections between the nodes. Rather, in Ohara's system, while the user is not required to interconnect the function blocks or icons, substantial user input is required to specify the generated program, e.g., the function blocks/icons (graphical objects) and their behavior. For example, as Ohara describes:

> With the visual programming method provided by the present invention, in order to automatically generate a program of a PLC, ***the user executes the steps of selecting an output signal defining a behavior; selecting a behavior; selecting signals relevant to the behavior; setting behavioral rules; and confirming the behavior.*** For operations carried out to execute the steps, the use of various input units can be thought of. Graphical objects such as a button, a check box, a slide bar and an icon are operated mainly by clicking or dragging the mouse 21 of the input apparatus 2. On the other hand, a string of characters is entered mainly by using the keyboard 22 of the input apparatus 2.
>
> ***The user selects a graphical object***, that is, an object defining a behavior, by clicking the mouse 21. On the signal box 200, a graphical object representing a signal and the name of the graphical object are displayed as a pair. (*emphasis added*)

Clearly, direct user input selecting graphical objects, selecting behaviors, setting behavioral rules, etc., is required. More specifically, Ohara's graphical objects are not created automatically as claimed, e.g., without direct user input specifying the plurality of nodes.

Regarding Figure 17, Ohara nowhere describes Figure 17 as a graphical program, nor as being automatically created as claimed. Rather, as described in the cited text, "the component model window 160 is used for verification of setting of behavioral characteristics for all output signals globally on a macro basis", where "Pieces of information set for behavioral characteristics of all output signals are thus displayed on the component model window 160 globally on a macro basis in order to allow all the pieces of information to be displayed at the same time." Appellant thus respectfully submits that the Examiner's characterization of Ohara is incorrect, and is not supported by the cited text. Additionally, as noted above, Ohara's generated program includes no user interface portion, which is a necessary feature of Appellant's invention as claimed. Nor does Weeren teach or suggest these features of claim 10.

Thus, for at least these reasons, Appellant submits that Ohara and Weeren, taken singly or in combination, fail to teach or suggest all the features and limitations of claim 10.

Claim 11

In addition to the distinctions noted above with regard to claim 1, the cited art fails to teach "automatically creating one or more user interface objects in the graphical program, wherein the one or more user interface objects perform one or more of providing input to or displaying output from the graphical program".

In asserting that Ohara teaches these features, the Examiner cites col. 44:20-35, and Figure 36. However, the cited text actually reads:

> A name `icon list` is displayed inside the window shown in FIG. 36. Reference numeral 3601 is a layout diagram window displaying a layout diagram showing a layout of driving members to be selected by the user. The layout diagram window 3601 includes a layout area 3620. An icon selected from those displayed on the member selection window 3610 is then relocated at a position in the layout area 3620. The layout area 3620 corresponds to output pins of an actual programmable logic controller (PLC). The user is said to carry out work to virtually connect a driving member controlled by a program to a programmable logic controller (PLC).

> Icons marked with Y00, Y01, Y02, - - - , YNM in the layout area 3620 are each referred to hereafter as an output pin icon.

Nowhere does the cited text (or any other portion of Ohara) disclose automatically creating one or more *user interface objects* in the graphical program. In fact, as argued above, Ohara, being directed to creation of programs for PLCs, which do not in general include graphical user interfaces, does not disclose creation of user interface objects at all.

Rather, as the cited text and Figure clearly indicate, the output pins are those of the actual PLC (programmable logic controller), and are specifically *not* for providing input to or displaying output *from the graphical program*. The user interfaces described by Ohara are for the development environment, not the generated program.

Appellant thus respectfully submits that the Examiner's characterization of Ohara is incorrect, and is not supported by the cited text. Nor does Weeren teach or suggest these features of claim 11.

Thus, for at least these reasons, Appellant submits that Ohara and Weeren, taken singly or in combination, fail to teach or suggest all the features and limitations of claim 11.

Claim 12, 44

In addition to the distinctions noted above with regard to claim 1, the cited art fails to teach "wherein the user input received specifies an instrumentation function; wherein the automatically generated graphical program implements the specified instrumentation function".

In asserting that Ohara teaches these features, the Examiner cites col. 65:50-56. However, the cited text actually reads:

> INDUSTRIAL APPLICABILITY
> As described above, the visual programming method provided by the present invention and the system adopting the method are based on a technique of user interfacing that allows a program to be generated automatically by selecting an object for defining a behavior, defining a behavior and setting a behavioral rule on a graphical editor.

Nowhere does the cited text (or any other portion of Ohara) disclose specifying an instrumentation function, nor automatically generated a graphical program that implements the specified instrumentation function. In fact, Ohara fails to mention instrumentation at all. Nor does Weeren teach or suggest these features of claim 12.

Thus, for at least these reasons, Appellant submits that Ohara and Weeren, taken singly or in combination, fail to teach or suggest all the features and limitations of claim 12.

Claim 13, 45

In addition to the distinctions noted above with regard to claim 1, the cited art fails to teach "wherein the instrumentation function comprises one or more of: a test and measurement function; or an industrial automation function".

In asserting that Ohara teaches these features, the Examiner cites col. 65:50-56. However, nowhere does the cited text, quoted above with reference to claim 12, (or any other portion of Ohara) disclose a test and measurement function, or an industrial automation function, nor an instrumentation function at all, and more specifically fails to teach or suggest specifying, and automatically generated a graphical program that implements the specified instrumentation function, i.e., a test and measurement function; or an industrial automation function. In fact, both Ohara and Weeren fail to mention instrumentation, test and measurement, or industrial automation, at all.

Thus, for at least these reasons, Appellant submits that Ohara and Weeren, taken singly or in combination, fail to teach or suggest all the features and limitations of claim 13.

Claim 14

In addition to the distinctions noted above with regard to claim 1, the cited art fails to teach "wherein said automatically generating the graphical program comprises calling an application programming interface (API) enabling the programmatic generation of a graphical program".

In asserting that Ohara teaches these features, the Examiner cites col. 16:19-28. However, the cited text reads:

> When the user operates the user interface based on the visual programming method provided by the present invention, the visual programming system provided by the invention automatically generates a program. In the following description, only matters related to the user interface characterizing the visual programming method are explained, excluding automatic generation of a program because the automatic generation of a program is the same as the conventional system

Nowhere does the cited text (or any other portion of Ohara) disclose automatically generating a graphical program by calling an application programming interface (API). In fact, Ohara never even mentions APIs at all. Appellant respectfully submits that automatically creating a graphical program via an API is substantially different from creating a program in the manner of Ohara, e.g., via interactions with a user interface. As is well known in the art, an API generally includes a set of *callable* functions (e.g., by a software process) for achieving some task; in this case, a set of callable functions for automatically creating a graphical program. Nowhere does the cited text (or any other portion of Ohara) disclose an API for such a purpose. The only APIs referred to by Weeren are the actual functional interfaces of (user-defined) functions or sub-routines themselves. Nowhere does Weeren disclose an API for generating a graphical program.

Thus, for at least these reasons, Appellant submits that Ohara and Weeren, taken singly or in combination, fail to teach or suggest all the features and limitations of claim 14.

<u>Claim 15</u>

In addition to the distinctions noted above with regard to claim 1, the cited art fails to teach "wherein said automatically generating the graphical program comprises automatically requesting a server program to generate the graphical program".

In asserting that Ohara teaches these features, the Examiner cites col. 25:8-10. However, the cited text reads:

> A generated program is transferred to a PLC typically by way of the network connection apparatus 5 shown in FIG. 1.

Clearly, the cited text actually discloses deploying the already generated program to a PLC over a network. Nowhere does the cited text (or any other portion of Ohara) disclose automatically generating a graphical program by automatically requesting a server program to generate the graphical program. Nowhere does Weeren disclose such a network request to automatically generate a graphical program.

Thus, for at least these reasons, Appellant submits that Ohara and Weeren, taken singly or in combination, fail to teach or suggest all the features and limitations of claim 15.

Claims 52, 53

In addition to the distinctions noted above with regard to claim 1 (and 34), the cited art fails to teach "wherein the graphical user interface portion comprises a front panel, wherein the front panel comprises one or more controls and/or one or more indicators".

In asserting that Weeren teaches these features, the Examiner cites "VerifyAcct" 402 of Figure 4. However, the cited element (VerifyAcct 402) is not a "graphical user interface portion" at all, but rather is a program icon with multiple returns representing a user-defined subroutine, as clearly described in col. 5:47 – col. 6:4. In fact, the only mention Weeren makes of a graphical user interface (GUI) at all is regarding a GUI for the operating system (col. 4:47-49). Nor does Ohara disclose automatically generating such a user interface portion. Thus, Weeren (and Ohara) fails to disclose this feature of claim 52.

Thus, for at least these reasons, Appellant submits that Ohara and Weeren, taken singly or in combination, fail to teach or suggest all the features and limitations of claim 52.

Claims 17, 18

Claim 17 is separately patentable because the cited reference does not teach or suggest the limitations recited in this claim.

The Examiner admits that Ohara fails to teach that the graphical program comprises a user interface portion, as well as automatically generating the block diagram portion without direct user input specifying the plurality of nodes or connections between the nodes, but asserts that Weeren remedies this admitted deficiency. Appellant respectfully disagrees.

For example, the Examiner asserts that Weeren teaches that the (automatically generated) graphical program comprises a graphical user interface portion, citing "VerifyAcct" 402 of Figure 4. However, the cited element (VerifyAcct 402) is not a "graphical user interface portion" at all, but rather is a program icon with multiple returns representing a user-defined subroutine, as clearly described in col. 5:47 – col. 6:4. In fact, the only mention Weeren makes of a graphical user interface (GUI) at all is regarding a GUI for the operating system (col. 4:47-49). Thus, Weeren (and Ohara) fails to disclose this feature of claim 17. The Examiner also asserts that Weeren discloses automatically generating the block diagram portion without direct user input specifying the plurality of nodes or connections between the nodes, citing Figure 6 and col.7:20-30. Appellant respectfully submits that Weeren discloses automatically drawing "arrows" (i.e., connections) associated with new program icons added to the program flow by the user, but does not teach or suggest automatically generating the block diagram portion *without direct user input specifying the plurality of nodes*. For example, the cited text reads:

> Flowchart 600 in FIG. 6 illustrates the steps performed in an exemplary embodiment of the present invention. In step 601, a program flow is graphically represented as a plurality of icons connected by at least one arrow. Additional icons are added to the program flow in step 602. The added icons represent a subroutine having overwriteable components that have a functionality which is represented by second arrows that are added in step 603. Other icons can be added to the second arrows in step 604. In step 605, the graphical development program indicates whether the added subroutine has been overwritten.

Note that no mention is made of automatically selecting or specifying icons to add to the flow diagram. In fact, as described in col.2:47-59:

> ***A developer adds functionality to the program flow by adding icons on existing arrows.*** The graphical development environment automatically adjusts the program flow to include the new icon and its associated arrows.
>
> For example, if ***the developer drops an icon representing a loop somewhere between the start and end of the program flow***, the environment automatically inserts the icon into the flow at that point and draws an arrow representing the loop. Subsequently, ***the developer can add other language components within the loop merely by dropping those icons on the arrow representing the loop***. (*emphasis added*)

Thus, in Weeren's system, icons added to the program flow diagram are specifically selected by the user and added/dropped by the user onto arrows in the diagram. In other words, according to Weeren, the user specifically provides direct user input specifying the program icons to be added to the program flow diagram. Thus, Weeren (and Ohara, as admitted by the Examiner) fails to teach or suggest these features and limitations of claim 17.

Furthermore, Appellant notes that Ohara explicitly excludes automatic generation of a program from the description of Ohara's invention:

> "In the following description, only matters related to the user interface characterizing the visual programming method are explained, excluding automatic generation of a program ***because the automatic generation of a program is the same as the conventional system***."(col. 16, lines 26-28)

In other words, Ohara does not consider the automatic generation of the program in Ohara's system and method to be novel, and does not disclose how it is performed, other than to say it is performed according to prior art approaches. Moreover, the generated program of Ohara is never described as a graphical program comprising a plurality of interconnected nodes that visually represent the desired functionality of the program. In fact, Appellant notes that Ohara never even mentions a graphical program at all, but rather discloses graphical *programming*, meaning creating a program via graphical means, i.e., via a graphical user interface without hand-coding the program, which, in light of Ohara's silence regarding graphical programs, as well as Ohara's statement that the actual automatic generation of the program is the same as the conventional system, indicates that Ohara's generated program (for deployment to the PLC) is a text-based program.

Thus, Appellant further submits that Ohara fails to properly disclose automatic generation of a graphical program, and thus does not provide proper support for the 103 rejection. Similarly, as argued above, Weeren also fails to disclose the automatic generation of a graphical program as claimed, and so also does not provide proper support for the 103 rejection.

Moreover, Appellant submits that neither Ohara or Weeren provides a proper motivation to combine. Appellant notes that the motivation to combine Ohara and Weeren suggested by the Examiner, "so that a program can be automatically generated without user's manual connections between nodes" (citing col. 2:17-21) is simply a statement of perceived benefit of the alleged combination. For example, the cited text reads:

> Accordingly, there is a need in the art for a graphical development environment that automatically connects icons in accordance with the call flow functionality of the underlying language component.

Nowhere does Weeren suggest the desirability of automatically generating the nodes (icons) in the graphical program, i.e., without direct user input specifying the plurality of nodes or connections between the nodes. Nor does Ohara suggest the desirability of automatically generating a user interface portion of a graphical program, nor of automatically generating the block diagram portion without direct user input specifying the plurality of nodes or connections between the nodes. Thus, Appellant submits that the Examiner's attempt to combine Ohara and Weeren to make a section 103 rejection is improper.

Additionally, Appellant submits that even were Ohara and Weeren properly combinable, which Appellant argues they are not, the resulting combination would still not produce Appellant's invention as represented in claim 17.

Thus, for at least these reasons, Appellant submits that Ohara and Weeren, taken singly or in combination, fail to teach or suggest all the features and limitations of claim 17. Appellant notes that the above arguments are also applicable to the other independent claims grouped therewith.

<u>Claims 19, 20</u>

In addition to the distinctions noted above with regard to claim 18 (presented above with regard to claim 17), the cited art fails to teach "wherein the GPG program comprises a graphical programming development environment application".

In asserting that Ohara teaches these features, the Examiner cites col. 1:19-28. However, the cited text reads:

> Technical fields of the present invention include function block diagrams and visual programming tools based on a diagram widely used as a programming language of a programmable logic controller (PLC), and a work environment for carrying out programming by operating graphical objects such as characters and icons which are displayed on a screen and used in a graphical user interface (GUI) commonly known as a user interface between the user and a personal computer or a workstation serving as a graphical editor.

Clearly, the cited text does not disclose automatically generating a graphical program as claimed. Moreover, nowhere does the cited text (or any other portion of Ohara) disclose a graphical programming development environment application being capable of automatically generating a graphical program in the manner claimed. Similarly, nowhere does Weeren disclose a graphical programming development environment application capable of automatically generating a graphical program in this manner.

Thus, for at least these reasons, Appellant submits that Ohara and Weeren, taken singly or in combination, fail to teach or suggest all the features and limitations of claim 19.

<u>Claim 23</u>

Claim 23 is separately patentable because the cited reference does not teach or suggest the limitations recited in this claim.

The Examiner asserts that Ohara teaches "automatically including graphical source code associated with the node in the graphical program, wherein the automatically included graphical source code implements the desired functionality, and wherein the graphical source code comprises a plurality of interconnected nodes that visually represent the desired functionality", citing col. 25:5-8, and Figure 17.

However, the cited text actually reads:

> Like the conventional system, the first embodiment is of course provided with a means for generating a program or code even though this means is shown in none of the figures.

Furthermore, Appellant notes that Ohara explicitly excludes automatic generation of a program from the description of Ohara's invention:

> "In the following description, only matters related to the user interface characterizing the visual programming method are explained, excluding automatic generation of a program ***because the automatic generation of a program is the same as the conventional system***."(col. 16, lines 26-28)

In other words, as argued above, Ohara does not consider the automatic generation of the program in Ohara's system and method to be novel, and does not disclose how it is performed, other than to say it is performed according to prior art approaches. Thus, Appellant further submits that Ohara fails to properly disclose automatic generation of a graphical program, and thus does not provide proper support for the 103 rejection. Similarly, as argued above, Weeren also fails to disclose the automatic generation of a graphical program as claimed, and so also does not provide proper support for the 103 rejection.

The cited text does not disclose automatically generating *graphical* source code associated with the node in the graphical program...wherein the graphical source code comprises a plurality of interconnected nodes that visually represent the desired functionality. The generated source code of Ohara is never described as graphical source code comprising plurality of interconnected nodes that visually represent the desired functionality of the program. In fact, Appellant notes that Ohara never even mentions a graphical program at all, but rather discloses graphical *programming*, meaning creating a program via graphical means, i.e., via a graphical user interface without hand-coding the program, which, in light of Ohara's silence regarding graphical programs, as well as Ohara's statement that the actual automatic generation of the program is the same as the conventional system, indicates that Ohara's generated program (for deployment to the PLC) is a text-based program.

Regarding Figure 17, Ohara nowhere describes Figure 17 as a graphical program, nor as being automatically created as claimed. In other words, the diagram of Figure 17 is not described by Ohara as being *the generated graphical program.* Rather, as described in the cited text, "the component model window 160 is used for verification of setting of behavioral characteristics for all output signals globally on a macro basis", where "Pieces of information set for behavioral characteristics of all output signals are thus displayed on the component model window 160 globally on a macro basis in order to allow all the pieces of information to be displayed at the same time." Appellant thus respectfully submits that the Examiner's characterization of Ohara is incorrect, and is not supported by the cited text.

The Examiner admits that Ohara fails to teach "automatically including the graphical source code as a sub-program of the graphical program without direct user input specifying the plurality of nodes or connections between the nodes, wherein the node represents the sub-program", but asserts that Weeren remedies this admitted deficiency, citing Figure 6 and col.7:20-30, and Figure 4. Appellant respectfully disagrees.

Appellant respectfully submits that, as argued above, Weeren discloses automatically drawing "arrows" (i.e., connections) associated with new program icons added to the program flow ***by the user,*** but does not teach or suggest automatically generating the including the graphical source code as a sub-program *without direct user input specifying the plurality of nodes or connections between the nodes*. For example, the cited text reads:

> Flowchart 600 in FIG. 6 illustrates the steps performed in an exemplary embodiment of the present invention. In step 601, a program flow is graphically represented as a plurality of icons connected by at least one arrow. Additional icons are added to the program flow in step 602. The added icons represent a subroutine having overwriteable components that have a functionality which is represented by second arrows that are added in step 603. Other icons can be added to the second arrows in step 604. In step 605, the graphical development program indicates whether the added subroutine has been overwritten.

Note that no mention is made of automatically selecting or specifying icons to add to the flow diagram. In fact, as described in col.2:47-59:

> ***A developer adds functionality to the program flow by adding icons on existing arrows.*** The graphical development environment automatically adjusts the program flow to include the new icon and its associated arrows.
>
> For example, if ***the developer drops an icon representing a loop somewhere between the start and end of the program flow,*** the environment automatically inserts the icon into the flow at that point and draws an arrow representing the loop. Subsequently, ***the developer can add other language components within***

> ***the loop merely by dropping those icons on the arrow representing the loop.*** (*emphasis added*)

Thus, in Weeren's system, icons added to the program flow diagram are specifically selected by the user and added/dropped by the user onto arrows in the diagram. In other words, according to Weeren, the user specifically provides direct user input specifying the program icons to be added to the program flow diagram. Thus, Weeren (and Ohara, as admitted by the Examiner) fails to teach or suggest these features and limitations of claim 23.

Regarding Figure 4, nowhere are the sub-routines represented iconically in Weeren's Figure 4 (or any other Figure of Weeren) described as being or automatically generated graphical source code comprising a plurality of interconnected nodes that visually represent the desired functionality.

Moreover, Appellant submits that neither Ohara or Weeren provides a proper motivation to combine. Appellant notes that the motivation to combine Ohara and Weeren suggested by the Examiner, "so that a program can be automatically generated without user's manual connections between nodes" (citing col. 2:17-21) is simply a statement of perceived benefit of the alleged combination. For example, the cited text reads:

> Accordingly, there is a need in the art for a graphical development environment that automatically connects icons in accordance with the call flow functionality of the underlying language component.

Nowhere does Weeren suggest the desirability of automatically generating the nodes (icons) in the graphical program, i.e., without direct user input specifying the plurality of nodes or connections between the nodes. Nor does Ohara suggest the desirability of automatically generating a user interface portion of a graphical program, nor of automatically generating the block diagram portion without direct user input specifying the plurality of nodes or connections between the nodes. Thus, Appellant submits that the Examiner's attempt to combine Ohara and Weeren to make a section 103 rejection is improper.

Additionally, Appellant submits that even were Ohara and Weeren properly combinable, which Appellant argues they are not, the resulting combination would still not produce Appellant's invention as represented in claim 23.

Thus, for at least these reasons, Appellant submits that Ohara and Weeren, taken singly or in combination, fail to teach or suggest all the features and limitations of claim 23. Appellant notes that the above arguments are also applicable to any other independent claims grouped therewith.

<u>Claim 26, 46</u>

Claim 26 is separately patentable because the cited reference does not teach or suggest the limitations recited in this claim.

The Examiner asserts that Ohara teaches "automatically ***including*** graphical source code associated with the node in the graphical program, wherein the automatically included graphical source code implements the desired functionality, and wherein the graphical source code comprises a plurality of interconnected nodes that visually represent the desired functionality", citing col. 25:5-8, and Figure 17. However, claim 26 recites "automatically ***generating*** graphical source code associated with the node to implement the specified functionality, wherein the graphical source code comprises a plurality of interconnected nodes that visually represent the desired functionality; wherein said automatically generating graphical source code associated with the node comprises automatically generating the graphical source code as a sub-program of the graphical program without direct user input specifying the plurality of nodes or connections between the nodes, wherein the node represents the sub-program.

Appellant submits that the arguments presented above with respect to independent claim 23 are equally applicable to claim 26, with the added distinction that in claim 26, the graphical source code associated with the node (comprising a plurality of interconnected nodes that visually represent the desired functionality) and included as a sub-program of the graphical program is also automatically generated, i.e., without direct user input specifying the plurality of nodes or connections between the nodes, wherein the node represents the sub-program.

Thus, for at least the reasons provided above, Appellant submits that Ohara and Weeren, taken singly or in combination, fail to teach or suggest all the features and limitations of claim 26. Appellant notes that the above arguments are also applicable to any other independent claims grouped therewith.

Claims 29, 49

In addition to the distinctions noted above with regard to claim 26, the cited art fails to teach "wherein no graphical source code is associated with the node until after said automatically generating graphical source code associated with the node".

In asserting that Ohara teaches these features, the Examiner simply asserts that "it is inherent that Ohara teaches no graphical source code or no functionality or no program instructions is associated with the node until after said programmatically generating graphical source code associated with the node. However, Appellant notes that this is simply speculation on the part of the Examiner, since neither reference describes this feature. Moreover, as noted above, nowhere does Ohara even mention graphical source code, nor automatically generating such in the automatic generation of a program, as claimed.

Thus, for at least these reasons, Appellant submits that Ohara and Weeren, taken singly or in combination, fail to teach or suggest all the features and limitations of claim 29.

Claims 30, 50

In addition to the distinctions noted above with regard to claim 26, the cited art fails to teach "wherein default graphical source code is associated with the node; wherein said automatically generating graphical source code associated with the node comprises replacing the default graphical source code with the automatically generated graphical source code".

In asserting that Ohara teaches these features, the Examiner cites col. 52:3-7, which read:

> All icons representing behavioral characteristics and behavioral conditions have default parameters stored internally so that a program created to operate in accordance with default parameters can also be checked as well.

Appellant notes that the cited text discloses default parameters, and nowhere teaches or suggests default graphical source code being associated with the node. In fact, as noted above, Ohara never even mentions graphical source code at all, and particularly fails to disclose such default graphical source code being associated with a node. Nor does Ohara disclose automatically generating graphical source code associated with the node including replacing the default graphical source code with the automatically generated graphical source code. Nor does Weeren disclose these features of claim 30.

Thus, for at least these reasons, Appellant submits that Ohara and Weeren, taken singly or in combination, fail to teach or suggest all the features and limitations of claim 30.

Claims 31

In addition to the distinctions noted above with regard to claim 26, the cited art fails to teach "wherein no functionality is defined for the node until after said automatically generating graphical source code associated with the node".

In asserting that Ohara teaches these features, the Examiner simply asserts that "it is inherent that Ohara teaches no graphical source code or no functionality or no program instructions is associated with the node until after said programmatically generating graphical source code associated with the node. However, Appellant notes that this is simply speculation on the part of the Examiner, since neither reference describes this feature. Moreover, Appellant notes that Ohara never even mentions graphical source code at all, and particularly fails to disclose automatically generating graphical source code in the manner claimed. Nor does Weeren teach these features of claim 31.

Thus, for at least these reasons, Appellant submits that Ohara and Weeren, taken singly or in combination, fail to teach or suggest all the features and limitations of claim 31.

Claim 32

In addition to the distinctions noted above with regard to claim 26, the cited art fails to teach "wherein no program instructions to be executed during execution of the graphical program are associated with the node until after said automatically generating graphical source code associated with the node".

In asserting that Ohara teaches these features, the Examiner simply asserts that "it is inherent that Ohara teaches no graphical source code or no functionality or no program instructions is associated with the node until after said programmatically generating graphical source code associated with the node. However, as noted above with regard to claims 29 and 31, this is simply speculation on the part of the Examiner, since neither reference describes this feature. Moreover, Appellant notes that Ohara never even mentions graphical source code at all, and particularly fails to disclose automatically generating graphical source code in the manner claimed. Nor does Weeren teach these features of claim 32.

Thus, for at least these reasons, Appellant submits that Ohara and Weeren, taken singly or in combination, fail to teach or suggest all the features and limitations of claim 32.

Claim 33, 51

In addition to the distinctions noted above with regard to claim 26, the cited art fails to teach "receiving user input requesting to change functionality of the node, after said automatically generating the graphical source code; re-displaying the GUI in response to the user input requesting to change functionality of the node; receiving user input to the GUI specifying new functionality for the node; automatically replacing the previously generated graphical source code with new graphical source code to implement the new functionality for the node".

In asserting that Ohara teaches these features, the Examiner cites col. 5:47-53, col. 16:20-29, Figure 4, and col. 25:5-8, which respectively read:

> In addition, in the visual programming method according to the present invention, the step of generating a behavioral rule not set yet from already set behavioral rules further includes a step of allowing a user to modify a generated behavioral rule by entering an acceptance or refusal input in response to the displayed behavioral rule. As a result, programming is made simple
>
> . . .
>
> When the user operates the user interface based on the visual programming method provided by the present invention, the visual programming system provided by the invention automatically generates a program. In the following

description, only matters related to the user interface characterizing the visual programming method are explained, excluding automatic generation of a program because the automatic generation of a program is the same as the conventional system.

...

Like the conventional system, the first embodiment is of course provided with a means for generating a program or code even though this means is shown in none of the figures.

Appellant notes that the cited text in no way discloses such editing of a node in a graphical program as claimed. Rather, the first citation describes user modification of a generated behavioral rule; the second citation simply states that Ohara does not disclose a new method for automatically generating a program, but that the program is automatically generated by conventional means; and the third citation reiterates this point of the second citation. Moreover, as noted above, Ohara nowhere even mentions graphical source code, and particularly fails to disclose automatically generating graphical source code in the manner claimed, and so does not and cannot support the Examiner's assertion. Nor does Weeren teach these features of claim 33.

Thus, for at least these reasons, Appellant submits that Ohara and Weeren, taken singly or in combination, fail to teach or suggest all the features and limitations of claim 33.

VIII. CONCLUSION

For the foregoing reasons, it is submitted that the Examiner's rejection of claims 1-7, 10-23, 26, 29-38, 40, 43-46, and 49-53 was erroneous, and reversal of his decision is respectfully requested.

The Commissioner is authorized to charge the appeal brief fee of $500.00 and any other fees that may be due to Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C. Deposit Account No. 501505/5150-48300/JCH. This Appeal Brief is submitted with a return receipt postcard.

Respectfully submitted,

[illegible]

Jeffrey C. Hood
Reg. No. 35,198
ATTORNEY FOR APPLICANT(S)

Meyertons Hood Kivlin Kowert & Goetzel, P.C.
P.O. Box 398
Austin, TX 78767-0398
Phone: (512) 853-8800

Date: 11/15/2005 JCH/MSW

IX. CLAIMS APPENDIX

The following lists claims 1-53 as incorporating entered amendments, including claims 1-7, 10-23, 26, 29-38, 40, 43-46, and 49-53 as on appeal.

1. (Previously Presented) A computer-implemented method for programmatically generating a graphical program, the method comprising:

displaying a graphical user interface (GUI) on a display;

receiving user input to the GUI specifying desired functionality of the graphical program; and

automatically generating the graphical program in response to the user input specifying the functionality of the graphical program, wherein the graphical program implements the specified functionality;

wherein the graphical program comprises a block diagram portion comprising a plurality of interconnected nodes, and a graphical user interface portion, wherein the plurality of interconnected nodes visually indicate functionality of the graphical program; and

wherein said automatically generating the graphical program includes generating the block diagram portion without direct user input specifying the plurality of nodes or connections between the nodes.

2. (Original) The method of claim 1,

wherein the GUI comprises information useable in guiding a user in creation of a program.

3. (Original) The method of claim 1,

wherein the GUI comprises one or more GUI input panels;

wherein the user input to the GUI comprises user input to each of the one or more GUI input panels.

4. (Original) The method of claim 3,

wherein said displaying the GUI and said receiving user input to the GUI comprise:

displaying a first GUI input panel on the display, wherein the first GUI input panel includes one or more first fields adapted to receive user input specifying first functionality of the graphical program;

receiving first user input specifying first functionality of the graphical program;

displaying a second GUI input panel on the display, wherein the second GUI input panel includes one or more second fields adapted to receive user input specifying second functionality of the graphical program;

receiving second user input specifying second functionality of the graphical program.

5. (Original) The method of claim 4,

wherein the second GUI input panel is one of a plurality of possible second GUI input panels, wherein the second GUI input panel is displayed based on the first user input.

6. (Previously Presented) The method of claim 1,

wherein said automatically generating the graphical program comprises automatically generating a portion of a graphical program.

7. (Previously Presented) The method of claim 1,

wherein said automatically generating the graphical program creates the graphical program without any user input specifying the new graphical program during said creating.

8. (Cancelled).

9. (Cancelled)

10. (Previously Presented) The method of claim 1, wherein said automatically generating the graphical program comprises:

automatically creating a plurality of graphical program objects in the graphical program; and

automatically interconnecting the plurality of graphical program objects in the graphical program;

wherein the interconnected plurality of graphical program objects comprise at least a portion of the graphical program.

11. (Previously Presented) The method of claim 1, wherein said automatically generating the graphical program comprises:

automatically creating one or more user interface objects in the graphical program, wherein the one or more user interface objects perform one or more of providing input to or displaying output from the graphical program.

12. (Previously Presented) The method of claim 1,

wherein the user input received specifies an instrumentation function;

wherein the automatically generated graphical program implements the specified instrumentation function.

13. (Original) The method of claim 12,

wherein the instrumentation function comprises one or more of:

a test and measurement function; or

an industrial automation function.

14. (Previously Presented) The method of claim 1,

wherein said automatically generating the graphical program comprises calling an application programming interface (API) enabling the programmatic generation of a graphical program.

15. (Previously Presented) The method of claim 1,

wherein said automatically generating the graphical program comprises automatically requesting a server program to generate the graphical program.

16. (Previously Presented) A computer-implemented method for automatically generating a graphical program, the method comprising:

displaying a plurality of GUI input panels on a display, wherein the GUI input panels comprise information useable in guiding a user in creation of a program;

receiving user input to the plurality of GUI input panels, wherein the user input specifies desired functionality of the graphical program; and

automatically generating the graphical program in response to the user input specifying the functionality of the graphical program, wherein the graphical program implements the specified functionality;

wherein the graphical program comprises a block diagram portion comprising a plurality of interconnected nodes, and a graphical user interface portion, wherein the plurality of interconnected nodes visually indicate functionality of the graphical program; and

wherein said automatically generating the graphical program includes generating the block diagram portion without direct user input specifying the plurality of nodes or connections between the nodes.

17. (Previously Presented) A computer-implemented method for automatically generating a graphical program, the method comprising:

displaying a graphical user interface (GUI) on a display;

receiving user input to the GUI specifying desired functionality of the graphical program;

executing a graphical program generation (GPG) program;

the GPG program receiving the user input, wherein the user input specifies the desired functionality of the new graphical program; and

the GPG program automatically generating the graphical program in response to the user input specifying the functionality of the graphical program, wherein the graphical program implements the specified functionality;

wherein the graphical program comprises a block diagram portion comprising a plurality of interconnected nodes, and a graphical user interface portion, wherein the plurality of interconnected nodes visually indicate functionality of the graphical program; and

wherein said automatically generating the graphical program includes generating the block diagram portion without direct user input specifying the plurality of nodes or connections between the nodes.

18. (Previously Presented) A computer-implemented method for automatically generating a graphical program, the method comprising:

displaying a graphical user interface (GUI) on a display;

receiving user input to the GUI indicating desired program operation of the graphical program;

executing a graphical program generation (GPG) program;

the GPG program receiving the user input, wherein the user input indicates desired operation of the graphical program; and

the GPG program automatically generating the graphical program in response to the user input indicating the desired operation of the graphical program, wherein the graphical program implements the desired operation;

wherein the graphical program comprises a block diagram portion comprising a plurality of interconnected nodes, and a graphical user interface portion, wherein the plurality of interconnected nodes visually indicate functionality of the graphical program; and

wherein said automatically generating the graphical program includes generating the block diagram portion without direct user input specifying the plurality of nodes or connections between the nodes.

19. (Original) The method of claim 18,

wherein the GPG program comprises a graphical programming development environment application.

20. (Original) The method of claim 18,

wherein the GPG program is operable to generate a plurality of graphical programs, depending on the received user input.

21. (Previously Presented) A computer-implemented method for automatically generating a graphical program, the method comprising:

displaying one or more input panels on a display;

receiving user input to the one or more input panels; and

automatically generating graphical source code for the graphical program, based on the received user input;

wherein the graphical program comprises a block diagram portion comprising a plurality of interconnected nodes, and a graphical user interface portion, wherein the plurality of interconnected nodes visually indicate functionality of the graphical program; and

wherein said automatically generating graphical source code for the graphical program includes generating graphical source code for the block diagram portion without direct user input specifying the plurality of nodes or connections between the nodes.

22. (Previously Presented) The method of claim 21,

wherein the one or more input panels comprise a graphical user interface (GUI) useable in guiding a user in specifying program functionality;

wherein the received user input specifies desired functionality of the graphical program;

wherein the automatically generated graphical source code implements the specified desired functionality.

23. (Previously Presented) A computer-implemented method for automatically generating a graphical program, the method comprising:

displaying a node in the graphical program in response to user input;

displaying a graphical user interface (GUI) for configuring functionality for the node in response to user input;

receiving user input via the GUI indicating desired functionality for the node; and

automatically including graphical source code associated with the node in the graphical program, wherein the automatically included graphical source code implements the desired functionality, and wherein the graphical source code comprises a plurality of interconnected nodes that visually represent the desired functionality;

wherein said automatically including graphical source code associated with the node in the graphical program comprises automatically including the graphical source code as a sub-program of the graphical program without direct user input specifying the plurality of nodes or connections between the nodes, wherein the node represents the sub-program.

24. (Cancelled)

25. (Cancelled)

26. (Previously Presented) A method for configuring a node in a graphical program, the method comprising:

displaying the node in the graphical program;

displaying a graphical user interface (GUI) associated with the node, wherein the GUI comprises information useable in guiding a user in specifying desired functionality for the node;

receiving user input to the GUI specifying desired functionality for the node; and

automatically generating graphical source code associated with the node to implement the specified functionality, wherein the graphical source code comprises a plurality of interconnected nodes that visually represent the desired functionality;

wherein said automatically generating graphical source code associated with the node comprises automatically generating the graphical source code as a sub-program of the graphical program without direct user input specifying the plurality of nodes or connections between the nodes, wherein the node represents the sub-program.

27. (Cancelled)

28. (Cancelled)

29. (Previously Presented) The method of claim 26,

wherein no graphical source code is associated with the node until after said automatically generating graphical source code associated with the node.

30. (Previously Presented) The method of claim 26,

wherein default graphical source code is associated with the node;

wherein said automatically generating graphical source code associated with the node comprises replacing the default graphical source code with the automatically generated graphical source code.

31. (Previously Presented) The method of claim 26,

wherein no functionality is defined for the node until after said automatically generating graphical source code associated with the node.

32. (Previously Presented) The method of claim 26,

wherein no program instructions to be executed during execution of the graphical program are associated with the node until after said automatically generating graphical source code associated with the node.

33. (Previously Presented) The method of claim 26, further comprising:

receiving user input requesting to change functionality of the node, after said automatically generating the graphical source code;

re-displaying the GUI in response to the user input requesting to change functionality of the node;

receiving user input to the GUI specifying new functionality for the node;

automatically replacing the previously generated graphical source code with new graphical source code to implement the new functionality for the node.

34. (Previously Presented) A memory medium for automatically generating a graphical program, the memory medium comprising program instructions executable to:

display a graphical user interface (GUI) on a display;

receive user input to the GUI specifying desired functionality of the graphical program; and

automatically generate the graphical program in response to the user input specifying the functionality of the graphical program, wherein the graphical program implements the specified functionality;

wherein the graphical program comprises a block diagram portion comprising a plurality of interconnected nodes, and a graphical user interface portion, wherein the plurality of interconnected nodes visually indicate functionality of the graphical program; and

wherein said automatically generating the graphical program includes generating the block diagram portion without direct user input specifying the plurality of nodes or connections between the nodes.

35. (Original) The memory medium of claim 34,

wherein the GUI comprises information useable in guiding a user in creation of a program.

36. (Original) The memory medium of claim 34,

wherein the GUI comprises one or more GUI input panels;

wherein the user input to the GUI comprises user input to each of the one or more GUI input panels.

37. (Original) The memory medium of claim 36, wherein the program instructions executable to display the GUI and the program instructions executable to receive user input to the GUI comprise program instructions executable to:

display a first GUI input panel on the display, wherein the first GUI input panel includes one or more first fields adapted to receive user input specifying first functionality of the graphical program;

receive first user input specifying first functionality of the graphical program;

display a second GUI input panel on the display, wherein the second GUI input panel includes one or more second fields adapted to receive user input specifying second functionality of the graphical program;

receive second user input specifying second functionality of the graphical program.

38. (Original) The memory medium of claim 37,

wherein the second GUI input panel is one of a plurality of possible second GUI input panels, wherein the second GUI input panel is displayed based on the first user input.

39. (Cancelled).

40. (Previously Presented) The memory medium of claim 34,

wherein said automatically generating the graphical program creates the graphical program without any user input specifying the new graphical program during said creating.

41. (Cancelled).

42. (Cancelled)

43. (Previously Presented) The memory medium of claim 34, wherein the program instructions executable to automatically generate the graphical program comprise program instructions executable to:

automatically create a plurality of graphical program objects in the graphical program; and

automatically interconnect the plurality of graphical program objects in the graphical program;

wherein the interconnected plurality of graphical program objects comprise at least a portion of the graphical program.

44. (Previously Presented) The memory medium of claim 34,

wherein the user input received specifies an instrumentation function;

wherein the automatically generated graphical program implements the specified instrumentation function.

45. (Original) The memory medium of claim 44,

wherein the instrumentation function comprises one or more of:

a test and measurement function; or

an industrial automation function.

46. (Previously Presented) A memory medium for configuring a node in a graphical program, the memory medium comprising program instructions executable to:

display the node in the graphical program;

display a graphical user interface (GUI) associated with the node, wherein the GUI comprises information useable in guiding a user in specifying desired functionality for the node;

receive user input to the GUI specifying desired functionality for the node; and automatically generate graphical source code associated with the node to implement the specified functionality, and wherein the graphical source code comprises a plurality of interconnected nodes that visually represent the desired functionality;

wherein said automatically generating graphical source code associated with the node comprises automatically generating the graphical source code as a sub-program of the graphical program without direct user input specifying the plurality of nodes or connections between the nodes, wherein the node represents the sub-program.

47-48. (Cancelled)

49. (Previously Presented) The memory medium of claim 46,

wherein no graphical source code is associated with the node until after said automatically generating graphical source code associated with the node.

50. (Previously Presented) The memory medium of claim 46,

wherein default graphical source code is associated with the node;

wherein said automatically generating graphical source code associated with the node comprises replacing the default graphical source code with the automatically generated graphical source code.

51. (Previously Presented) The memory medium of claim 46, further comprising program instructions executable to:

receive user input requesting to change functionality of the node, after said automatically generating the graphical source code;

re-display the GUI in response to the user input requesting to change functionality of the node;

receive user input to the GUI specifying new functionality for the node;

automatically replace the previously generated graphical source code with new graphical source code to implement the new functionality for the node.

52. (Previously Presented) The method of claim 1, wherein the graphical user interface portion comprises a front panel, wherein the front panel comprises one or more controls and/or one or more indicators.

53. (Previously Presented) The memory medium of claim 34, wherein the graphical user interface portion comprises a front panel, wherein the front panel comprises one or more controls and/or one or more indicators.

X. EVIDENCE APPENDIX

No evidence submitted under 37 CFR §§ 1.130, 1.131 or 1.132 or otherwise entered by the Examiner is relied upon in this appeal.

XI. RELATED PROCEEDINGS APPENDIX

There are no related proceedings.